

DESAIN APLIKASI LAUNDRY

Laporan Praktikum Pemrograman Berbasis Objek 5

Dosen Pengampu :

Nurfiah, S.ST, M.Kom



OLEH :

SHERLY SUKMADIRA PUTRI

2311532015

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

2024

BAB I

LATAR BELAKANG

A. Latar Belakang

Dalam pengelolaan bisnis laundry, pencatatan data pesanan pelanggan (Order Pelanggan) dan rincian pesanan (Order Detail) memiliki peran penting untuk memastikan layanan yang efisien dan terorganisir. Data ini mencakup informasi seperti identitas pelanggan, jenis layanan yang dipilih, jumlah item yang dicuci, serta biaya yang dikenakan.

Namun, pengelolaan data secara manual rentan terhadap kesalahan, kehilangan data, dan ketidakefektifan dalam proses pencatatan maupun pelacakan pesanan. Oleh karena itu, diperlukan sistem berbasis aplikasi yang mampu mengotomatisasi proses ini.

Praktikum ini dilakukan untuk mempelajari dan menerapkan konsep dasar dalam pengelolaan data pesanan menggunakan aplikasi berbasis Java Swing. Dengan mempraktikkan pembuatan form input, tombol aksi untuk menyimpan dan memperbarui data, serta pengolahan data pesanan, diharapkan mahasiswa dapat memahami cara merancang dan mengimplementasikan sistem pencatatan data yang efektif untuk mendukung operasional bisnis laundry.

B. Tujuan

Mengimplementasikan sistem pencatatan data pesanan pelanggan (Order Pelanggan) dan rincian pesanan (Order Detail) menggunakan aplikasi berbasis Java Swing.

C. Alat

Computer / laptop yang telah terinstall JDK dan Eclipse

BAB II

LANDASAN TEORI

A. Java Swing

Java Swing adalah salah satu library di Java yang digunakan untuk membuat antarmuka pengguna grafis (Graphical User Interface - GUI). Swing menyediakan berbagai komponen seperti tombol (JButton), label (JLabel), tabel (JTable), dan kotak teks (JTextField) yang mendukung pembuatan aplikasi berbasis desktop. Dalam praktikum ini, Java Swing digunakan untuk membangun antarmuka pengelolaan data pesanan dan rincian layanan laundry.

B. Pemrograman Berorientasi Objek (OOP)

OOP adalah paradigma pemrograman yang berbasis pada konsep objek dan kelas. Kelas merepresentasikan blueprint untuk objek, sedangkan objek adalah instance dari kelas tersebut. Dalam aplikasi ini, konsep OOP digunakan untuk merepresentasikan data pelanggan dan detail pesanan sebagai objek dengan atribut dan metode yang relevan.

C. Event Handling

Event Handling adalah mekanisme yang digunakan untuk menangani aksi pengguna, seperti klik tombol atau pemilihan item pada daftar. Praktikum ini memanfaatkan event handling melalui ActionListener untuk mendeteksi dan merespons aksi pada tombol tambah, ubah, hapus, dan batal.

D. Model-View-Controller (MVC)

MVC adalah pola desain yang memisahkan logika aplikasi menjadi tiga bagian:

- Model : Mengelola data dan logika bisnis, seperti data pesanan pelanggan.
- View : Menampilkan antarmuka pengguna dan data yang dimuat.
- Controller : Menghubungkan antara Model dan View serta menangani interaksi pengguna.

Dalam aplikasi ini, pola MVC digunakan untuk memastikan pemisahan logika dan tampilan agar lebih mudah dikelola dan dikembangkan.

E. JTable dan DefaultTableModel

JTable adalah komponen yang digunakan untuk menampilkan data dalam bentuk tabel. DefaultTableModel memudahkan pengelolaan data tabel dengan memungkinkan penambahan, penghapusan, dan pengubahan data secara dinamis. Pada praktikum ini, JTable digunakan untuk menampilkan daftar pesanan dan layanan laundry yang tersedia.

F. Perhitungan Dinamis dengan Listener

Aplikasi ini menerapkan fitur pengisian otomatis pada kolom harga dan total berdasarkan layanan yang dipilih dan jumlah pesanan. Proses ini memanfaatkan listener untuk mendeteksi perubahan input dan memperbarui nilai secara real-time.

G. Validasi Input dan Manajemen Data

Validasi input memastikan data yang dimasukkan operator sesuai dengan format yang diharapkan, seperti angka untuk jumlah pesanan dan harga. Data yang diinput juga disimpan dan dikelola dalam struktur terorganisir untuk kemudahan akses dan manipulasi.

Dengan mengacu pada konsep-konsep di atas, praktikum ini dirancang untuk membantu mahasiswa memahami penerapan fitur-fitur tersebut dalam pembuatan aplikasi laundry berbasis Java.

BAB III

PROSEDUR DAN PENGAPLIKASIAN

Terdapat beberapa Langkah untuk mencapai tujuan dari praktikum ini diantaranya :

1. Membuat JDialog Pelanggan

JDialog berfungsi untuk menampilkan nama-nama pelanggan yang ada pada aplikasi, Ketika operator klik form pelanggan maka akan ditampilkan daftar pelanggan, selanjutnya operator memilih pelanggan maka secara otomatis form pelanggan akan terisi oleh data pelanggan tersebut.

Ada 2 buah parameter yang dikirimkan oleh JDialog Ketika memilih pelanggan yaitu id dan nama, id merupakan id pelanggan yang nantinya akan disimpan sementara pada variable `id_pelanggan`, kemudian nama akan ditampilkan pada form nama, berikut ini Langkah-langkahnya.

- Buat package baru dengan nama listener kemudian tambahkan file java baru dengan nama `DataListener.java` file ini akan digunakan sebagai interface yang berfungsi menangkap data yang dikirimkan dari JDialog Pelanggan, tambahkan kode program dibawah ini.

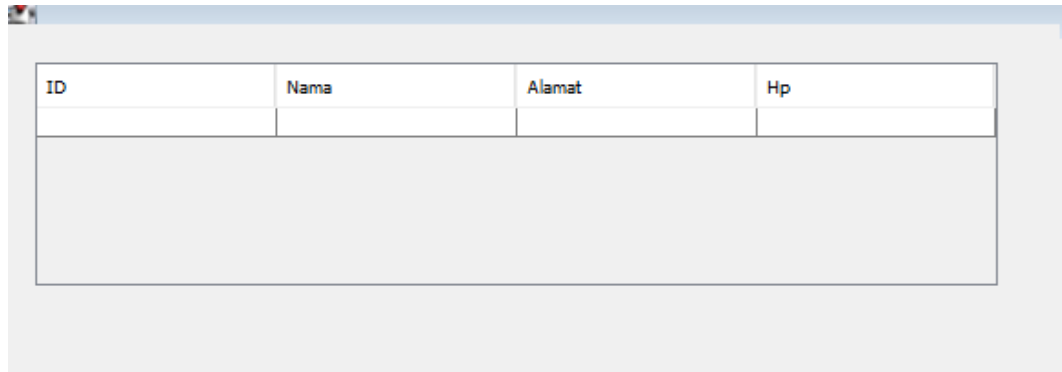


```

Welcome.java login.java create_acco... home.java Report.java pr
1 package listener;
2
3 public interface DataListener {
4     void onDataReceived(String id, String nama);
5 }
6

```

- Tambahkan JDialog pada package ui dengan nama DialogPelanggan.java, buat desain UI seperti gambar berikut :



- Tambahkan kode program berikut pada DialogPelanggan.java

```

DialogPelanggan.java - Eclipse IDE
gate Search Project Run Window Help
Welcome.java login.java create_acco... home.java Report.java profile.java
1 package Ui;
2
3 import java.awt.EventQueue;
19
20 public class DialogPelanggan extends JFrame {
21     private static final long serialVersionUID = 1L;
22     private JPanel contentPane;
23     private JTable tableplgn;
24
25     private DataListener listener;
26     CustomerRepo usr = new CustomerRepo();
27     List<Customer> ls;
28     public String id;
29
30     public void loadTable() {
31         ls = usr.show();
32         if (ls != null) {
33             TableCustomer tu = new TableCustomer(ls);
34             tableplgn.setModel(tu);
35             tableplgn.getTableHeader().setVisible(true);
36         }
37     }
38

```

```

60 */
61 public DialogPelanggan(DataListener listener) {
62     this.Listener = listener;
63     setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
64     setBounds(100, 100, 600, 399);
65     contentPane = new JPanel();
66     contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
67
68     setContentPane(contentPane);
69     contentPane.setLayout(null);
70
71     JScrollPane scrollPane = new JScrollPane();
72     scrollPane.setBounds(23, 21, 529, 122);
73     contentPane.add(scrollPane);
74
75     tableplgn = new JTable();
76     tableplgn.addMouseListener(new MouseAdapter() {
77         @Override
78         public void mouseClicked(MouseEvent e) {
79             listener.onDataReceived(
80                 tableplgn.getValueAt(tableplgn.getSelectedRow(), 0).toString(),
81                 tableplgn.getValueAt(tableplgn.getSelectedRow(), 1).toString()
82             );
83             dispose();
84         }
85     });
86     tableplgn.setModel(new DefaultTableModel(
87         new Object[][] {
88             {null, null, null, null},
89         },
90         new String[] {
91             "ID", "Nama", "Alamat", "Hp"
92         }
93     ));
94     scrollPane.setViewportView(tableplgn);
95     loadTable();
96 }

```

The screenshot shows an IDE with multiple tabs. The active tab is 'DialogPela...', which displays the same Java code as the first block. The code defines a constructor for 'DialogPelanggan' that takes a 'DataListener' and sets up a 'JTable' with columns 'ID', 'Nama', 'Alamat', and 'Hp'. A 'MouseListener' is added to the table, which calls 'listener.onDataReceived' with the selected row's data and then 'dispose()'. The table is initially populated with one row of null values. The code also sets the window's close operation to 'DISPOSE_ON_CLOSE' and sets the content pane with a null layout.

- Menampilkan JDialog Ketika form pelanggan diklik, tambahkan event handler berikut pada txtPelanggan yang berada pada class OrderDetailFrame pada Ui.

```

txtPelanggan = new JTextField();
txtPelanggan.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        DialogPelanggan dialog = new DialogPelanggan(OrderDetailFrame.this);
        dialog.setVisible(true);
    }
});
txtPelanggan.setColumns(10);

```

- Tambahkan implements DataListener pada OrderDetailFrame

```

43
44 public class OrderDetailFrame extends JFrame implements DataListener {
45

```

- Maka akan meng-override method onDataReceived yang berfungsi menerima data id costumer dan nama costumer.

```

    public void onDataReceived(String id, String nama) {
        txtPelanggan.setText(nama);
        id_pelanggan=id;
    }

```

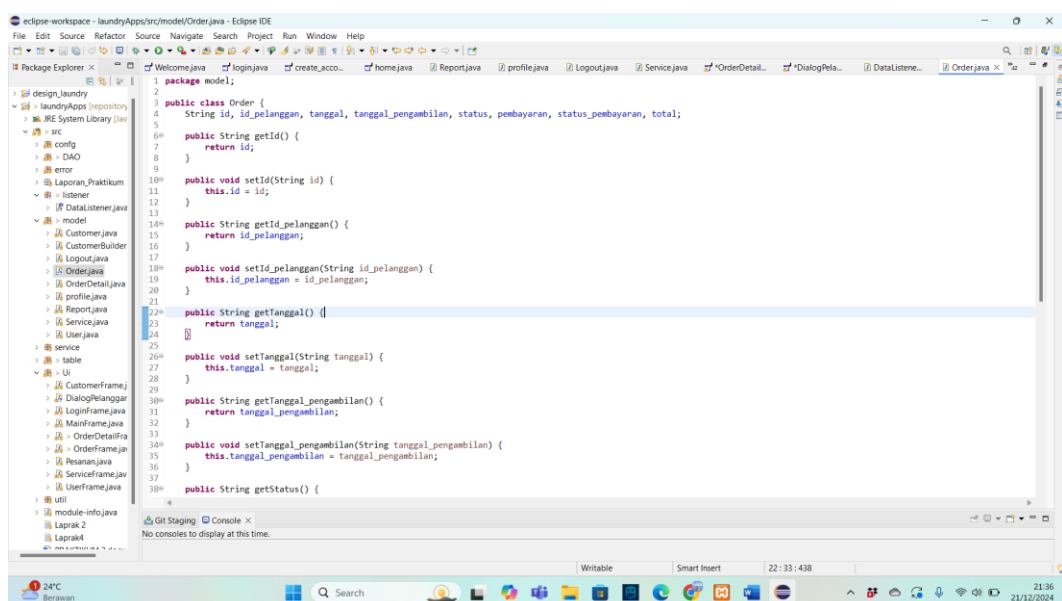
2. Menyimpan data order

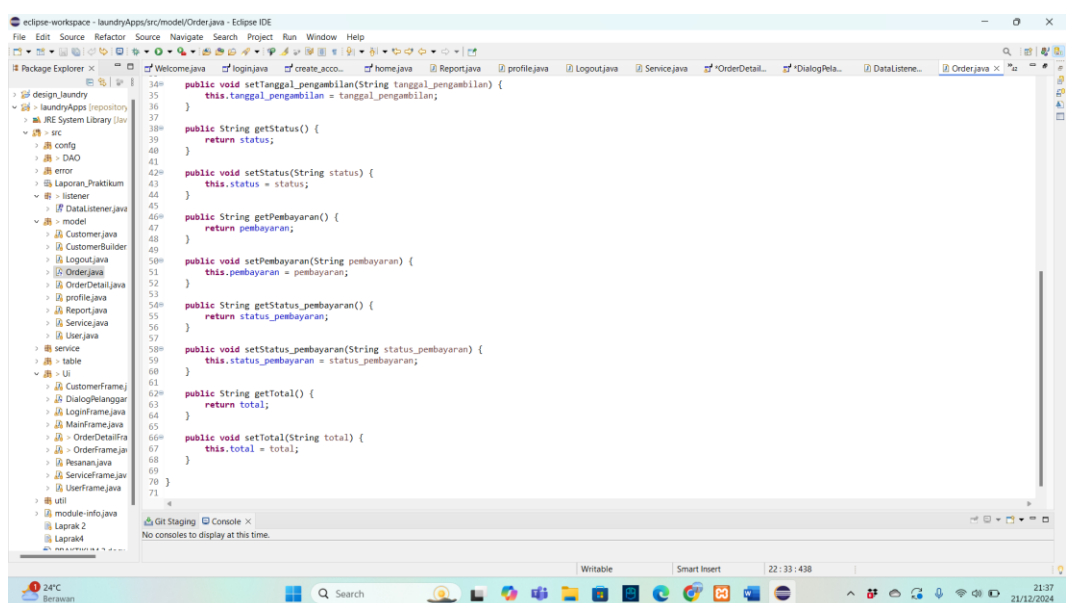
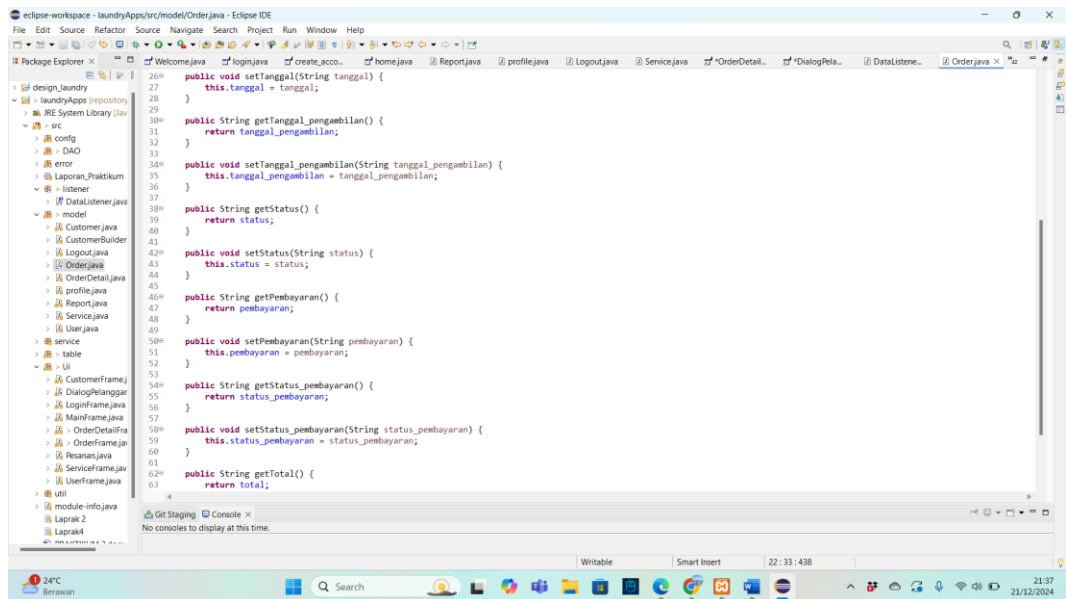
Setelah proses CRUD order detail selesai selanjutnya menyimpan data Order, berikut ini Langkahlangkahnya.

- Buat table baru dengan nama Orders dan Struktur Table seperti berikut

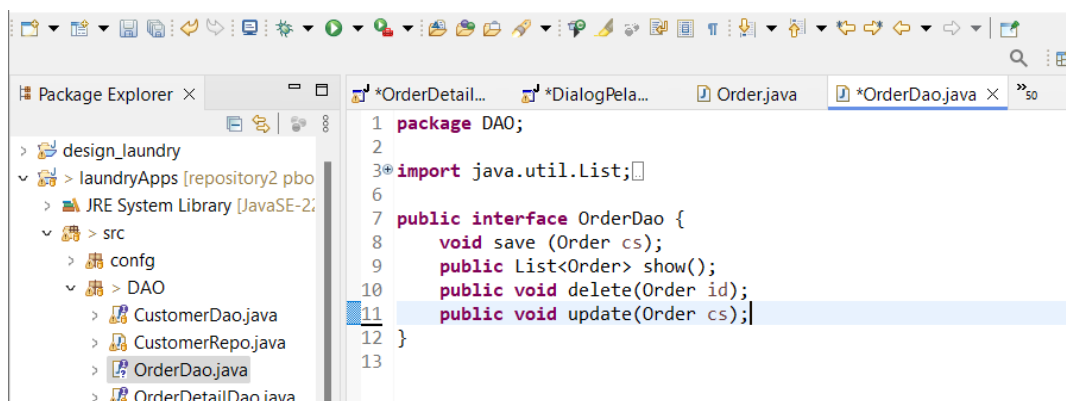
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 id	varchar(11)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	2 id_pelanggan	int(11)			No	None			Change Drop More
<input type="checkbox"/>	3 tanggal	date			No	None			Change Drop More
<input type="checkbox"/>	4 tanggal_pengambilan	date			No	None			Change Drop More
<input type="checkbox"/>	5 status	varchar(32)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	6 pembayaran	varchar(32)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	7 status_pembayaran	varchar(32)	utf8mb4_general_ci		No	None			Change Drop More
<input type="checkbox"/>	8 total	int(32)			No	None			Change Drop More

- Tambahkan model baru pada package model dengan nama Order dan tambahkan kode program beirkut.





- Tambahkan DAO pada package dao dengan nama OrderDao dan tambahkan kode program berikut.



- Tambahkan OrderRepo pada package dao dan tambahkan kode program berikut ini


```

DAO/OrderRepo.java - Eclipse IDE
: Navigate Search Project Run Window Help
Welcome.java login.java create_acco... home.java Report.java profile.java Logout.java Service.java User.java OrderFrame.java OrderRepo.java x
1 package DAO;
2
3 import java.util.logging.Level;
14
15 public class OrderRepo{
16
17     private Connection connection;
18     final String insert = "INSERT INTO orders (id, id_pelanggan, tanggal, tanggal_pengambilan, status, pembayaran, status_pembayaran, total)" + "VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
19     final String select = "SELECT * FROM orders;";
20     final String delete = "DELETE FROM orders WHERE id=?;";
21     final String delete_detail = "DELETE FROM order_detail WHERE order_id=?;";
22     final String update = "UPDATE orders SET id_pelanggan=?, tanggal=?, tanggal_pengambilan=?, status=?, pembayaran=?, status_pembayaran=?, total=? WHERE id=?;";
23
24
25
26 public OrderRepo() {
27     connection = Database.getInstance().getConnection();
28 }
29
30 public void save(Order cs) {
31     PreparedStatement st = null;
32     try {
33         // Inisialisasi PreparedStatement
34         st = connection.prepareStatement(insert);
35
36         // Set parameter untuk query SQL
37         st.setString(1, cs.getId());
38         st.setString(2, cs.getId_pelanggan());
39         st.setString(3, cs.getTanggal());
40         st.setString(4, cs.getTanggal_pengambilan());
41         st.setString(5, cs.getStatus());
42         st.setString(6, cs.getPembayaran());
43         st.setString(7, cs.getStatus_pembayaran());
44         st.setString(8, cs.getTotal());
45
46         // Eksekusi query
47         st.executeUpdate();
48

```

```

Welcome.java login.java create_acco... home.java Report.java profile.java Logout.java Service.java User.java OrderFrame.java OrderRepo.java x
47         st.executeUpdate();
48     } catch (SQLException e) {
49         e.printStackTrace();
50     } finally {
51         try {
52             // Pastikan `st` tidak null sebelum menutup
53             if (st != null) {
54                 st.close();
55             }
56         } catch (SQLException e) {
57             e.printStackTrace();
58         }
59     }
60 }
61
62
63
64 public List<Order> show(){
65     List<Order> ls=null;
66     try {
67         ls = new ArrayList<Order>();
68         Statement st = connection.createStatement();
69         ResultSet rs = st.executeQuery(select);
70         while(rs.next()) {
71             Order cs = new Order();
72             cs.setId(rs.getString("id"));
73             cs.setId_pelanggan(rs.getString("id_pelanggan"));
74             cs.setTanggal(rs.getString("tanggal"));
75             cs.setStatus(rs.getString("status"));
76             cs.setPembayaran(rs.getString("pembayaran"));
77             cs.setStatus_pembayaran(rs.getString("status_pembayaran"));
78             cs.setTotal(rs.getString("total"));
79             ls.add(cs);
80         }
81     } catch (SQLException e) {
82         Logger.getLogger(UserDao.class.getName()).log(Level.SEVERE, null, e);
83     }
84     return ls;

```

```
src/DAO/OrderRepo.java - Eclipse IDE
urce  Navigate  Search  Project  Run  Window  Help

Welcome.java  login.java  create_acco...  home.java  Report.java  profile.java  Logout.java  Service.java  User.java  OrderFrame.java  OrderRepo.java x

84      }
85      return ls;
86  }
87  public void delete(String id) {
88      PreparedStatement st = null;
89      PreparedStatement st_detail = null;
90      try {
91          st = connection.prepareStatement(delete);
92          st.setString(1, id);
93          st.executeUpdate();
94
95          st_detail = connection.prepareStatement(delete_detail);
96          st_detail.setString(1, id);
97          st_detail.executeUpdate();
98      } catch (SQLException e) {
99          e.printStackTrace();
100     } finally {
101         try {
102             st.close();
103             st_detail.close();
104         } catch (SQLException e) {
105             e.printStackTrace();
106         }
107     }
108 }
109
110 public void update(Order cs) {
111     PreparedStatement st = null;
112     try {
113         st = connection.prepareStatement(update);
114
115         st = connection.prepareStatement(update);
116         st.setString(1, cs.getId_pelanggan());
117         st.setString(2, cs.getTanggal());
118         st.setString(3, cs.getTanggal_pengambilan());
119         st.setString(4, cs.getStatus());
120         st.setString(5, cs.getPembayaran());
121         st.setString(6, cs.getStatus_pembayaran());
122     } catch (SQLException e) {
123         e.printStackTrace();
124     } finally {
125         try {
126             st.close();
127             st_detail.close();
128         } catch (SQLException e) {
129             e.printStackTrace();
130         }
131     }
132 }
133 }
134 }
135 }
```

```
Welcome.java  login.java  create_acco...  home.java  Report.java  profile.java  Logout.java  Service.java  User.java  OrderFrame.java  OrderRepo.java x

98     } catch (SQLException e) {
99         e.printStackTrace();
100     } finally {
101         try {
102             st.close();
103             st_detail.close();
104         } catch (SQLException e) {
105             e.printStackTrace();
106         }
107     }
108 }
109
110 public void update(Order cs) {
111     PreparedStatement st = null;
112     try {
113         st = connection.prepareStatement(update);
114
115         st = connection.prepareStatement(update);
116         st.setString(1, cs.getId_pelanggan());
117         st.setString(2, cs.getTanggal());
118         st.setString(3, cs.getTanggal_pengambilan());
119         st.setString(4, cs.getStatus());
120         st.setString(5, cs.getPembayaran());
121         st.setString(6, cs.getStatus_pembayaran());
122         st.setString(7, cs.getTotal());
123         st.setString(8, cs.getId());
124         st.executeUpdate();
125     } catch (SQLException e) {
126         e.printStackTrace();
127     } finally {
128         try {
129             st.close();
130         } catch (SQLException e) {
131             e.printStackTrace();
132         }
133     }
134 }
135 }
```

3. Sebelum menyimpan data pada Order, tambahkan kode program berikut pada tombol simpan yang berada pada Order di OrderDetailFrame.

```

JButton btnNewButton = new JButton("Simpan");
btnNewButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        OrderRepo order_repo = new OrderRepo();
        if(id_pelanggan != ""){
            Order order = new Order();
            order.setId(txtid_Order.getText());
            order.setId_pelanggan(id_pelanggan);
            order.setTanggal(txtTanggal.getText());
            order.setTanggal_pengambilan(txtTanggalPengambilan.getText());
            order.setStatus(cbStatus.getSelectedItem().toString());
            order.setStatus_pembayaran(cbStatusPembayaran.getSelectedItem().toString());
            order.setPembayaran(cbPembayaran.getSelectedItem().toString());
            order.setTotal(txtTotalOrder.getText());
            order_repo.save(order);
            JOptionPane.showMessageDialog(null, "Oder berhasil disimpan");
        }else {
            JOptionPane.showMessageDialog(null, "Silakan pilih pelanggan terlebih dahulu");
        }
    }
});
btnNewButton.setBounds(34, 583, 87, 35);
panel.add(btnNewButton);

```

4. Data pada Order sudah bisa di simpan pada GUI OrderDetail pada bagian kiri. Sebelum menyimpan data Order maka harus dipastikan terlebih dahulu data pelanggan, tanggal, tanggal pengambilan, status, pembayaran dan status pembayaran baru diklik tombol simpan.

The screenshot displays the OrderDetail GUI with the following components:

- Form Fields (Left Side):**
 - Oder ID:
 - Pelanggan:
 - Tanggal:
 - Tanggal Pengambilan:
 - Status:
 - Total:
 - Pembayaran:
 - Status Pembayaran:
- Buttons (Bottom Left):**
- Tables (Right Side):**
 - Layanan Table:**

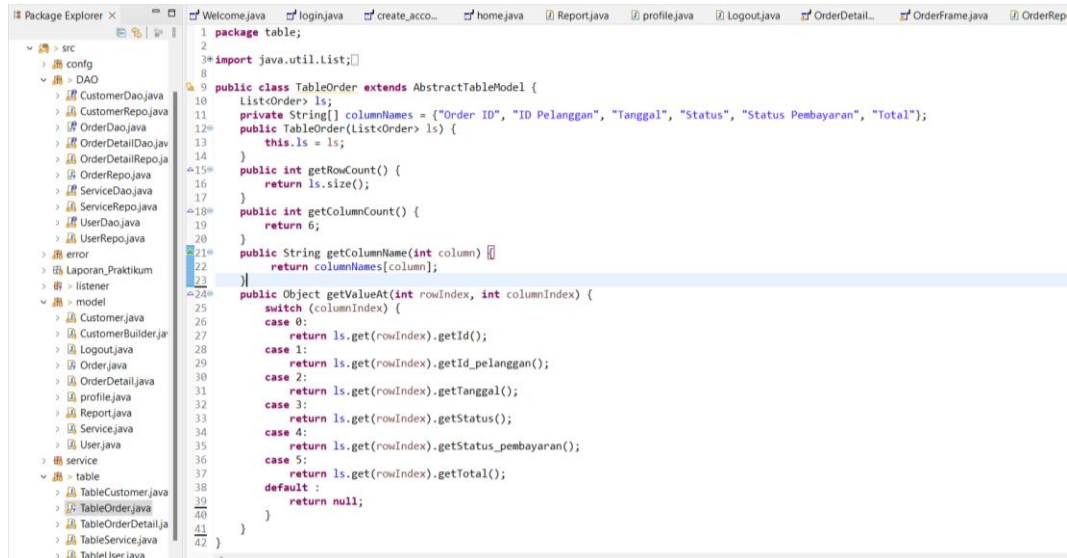
ID	Jenis	Status	Harga	Satuan
1	baju	proses	5000.0	3.0
 - Summary Fields:**
 - HargaKG:
 - Jenis:
 - Jumlah:
 - Total:
 - Buttons:**
 - Order Items Table:**

Id	Jenis	Quantity	Total
1	baju	2	10000.0
3	baju	2	10000.0

5. Menampilkan Data Order

Setelah data order berhasil disimpan, selanjutnya menampilkan data order dengan mengikuti Langkah kecil berikut.

- Pertama-tama buatlah terlebih dahulu class untuk table order, tambahkan class baru pada package table dengan nama TableOrder tambahkan kode program berikut :

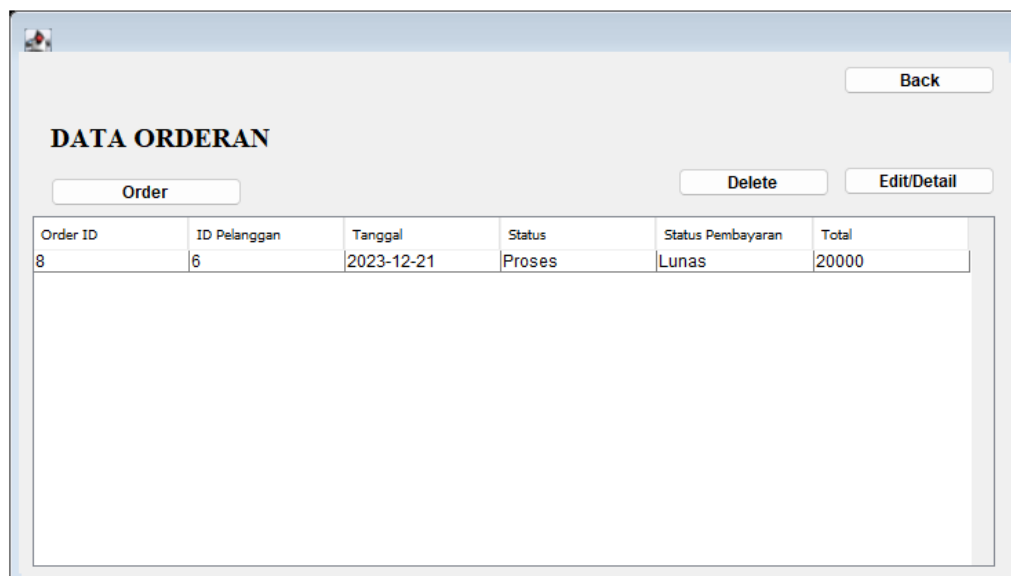


```

1 package table;
2
3 import java.util.List;
4
5 public class TableOrder extends AbstractTableModel {
6     List<Order> ls;
7     private String[] columnNames = {"Order ID", "ID Pelanggan", "Tanggal", "Status", "Status Pembayaran", "Total"};
8     public TableOrder(List<Order> ls) {
9         this.ls = ls;
10    }
11    public int getRowCount() {
12        return ls.size();
13    }
14    public int getColumnCount() {
15        return 6;
16    }
17    public String getColumnName(int column) {
18        return columnNames[column];
19    }
20    public Object getValueAt(int rowIndex, int columnIndex) {
21        switch (columnIndex) {
22            case 0:
23                return ls.get(rowIndex).getId();
24            case 1:
25                return ls.get(rowIndex).getId_pelanggan();
26            case 2:
27                return ls.get(rowIndex).getTanggal();
28            case 3:
29                return ls.get(rowIndex).getStatus();
30            case 4:
31                return ls.get(rowIndex).getStatus_pembayaran();
32            case 5:
33                return ls.get(rowIndex).getTotal();
34            default:
35                return null;
36        }
37    }
38 }

```

- Kemudian tambahkan JFrame baru dengan nama OrderFrame, buat desainnya seperti gambar berikut. *Perlu diingat bahwa kolom itu belum terisi jika kita belum melakukan save order pada OrderDetailFrame.



- Tambahkan kode program berikut pada OrderFrame

```
// Repository and data
private OrderRepo repo_od = new OrderRepo();
private List<Order> ls_od;
private String order_id = "";
```

- Buat method untuk menampilkan data order

```
134
135- /**
136     * Load data into the table.
137     */
138- public void loadTableOrder() {
139     ls_od = repo_od.show();
140     TableOrder tu = new TableOrder(ls_od);
141     table.setModel(tu);
142     table.getTableHeader().setVisible(true);
143 }
144 }
145
```

- Panggil method loadTableOrder() pada main frame

```
private static void main(String[] args) {
    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(() -> {
            try {
                OrderFrame frame = new OrderFrame();
                frame.setVisible(true);
            } catch (Exception e) {
                e.printStackTrace();
            }
        });
    }
}
```

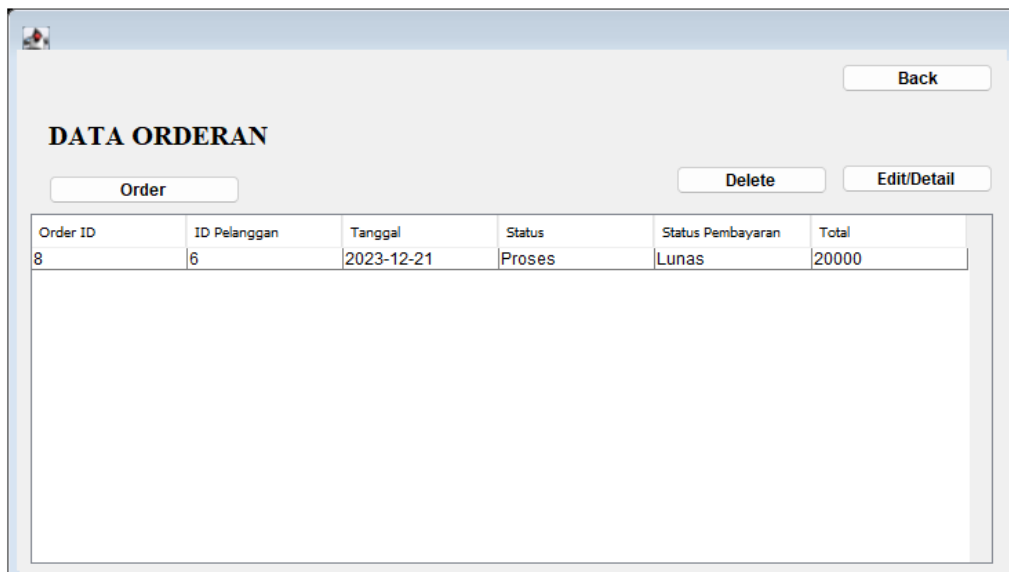
- Tambahkan kode program berikut pada event handler Buat Orderan sehingga akan tampil form OrderDetailFrame dan dapat menambahkan data order.

```
btnOrder.addActionListener(e -> {
    OrderDetailFrame odf = new OrderDetailFrame();
    odf.setVisible(true);
    odf.loadTableOrderDetail();
    odf.loadTableService();
    dispose();
});
```

- Tambahkan kode program berikut pada event handler pada tombol hapus.

```
// Event Listeners
btnDelete.addActionListener(e -> {
    if (!order_id.isEmpty()) {
        repo_od.delete(order_id);
        loadTableOrder(); // Refresh table after deletion
    } else {
        JOptionPane.showMessageDialog(null, "Pilih data yang akan di hapus");
    }
});
```

- Tampilan Data Order



The screenshot shows a Java Swing window titled "DATA ORDERAN". The window has a light gray background and a title bar. In the top right corner, there is a "Back" button. Below the title bar, there is a section with the title "DATA ORDERAN" in bold. Underneath, there are three buttons: "Order", "Delete", and "Edit/Detail". Below these buttons is a table with the following data:

Order ID	ID Pelanggan	Tanggal	Status	Status Pembayaran	Total
8	6	2023-12-21	Proses	Lunas	20000

Below the table, there is a large empty rectangular area, likely a placeholder for more data or a scrollable list.

BAB IV

SIMPULAN

Praktikum ini berhasil mengimplementasikan sistem pengelolaan data pesanan pelanggan dan detail pesanan laundry menggunakan Java Swing. Melalui proses ini, praktikan dapat memahami dan mempraktikkan konsep pemrograman berorientasi objek (OOP), pengelolaan event handling, serta penerapan komponen GUI seperti JButton, JTextField, dan JTable.

Aplikasi yang dibuat mampu menangani berbagai fungsi, seperti menambahkan, mengubah, dan menghapus data pesanan, serta menghitung total biaya secara otomatis berdasarkan jenis layanan dan jumlah pesanan yang diinput. Selain itu, penerapan validasi input memastikan keakuratan data yang dimasukkan oleh operator.

Dengan memanfaatkan pola desain Model-View-Controller (MVC), aplikasi ini juga memberikan struktur yang terorganisir, memudahkan proses pengembangan dan pemeliharaan ke depannya. Praktikum ini membuktikan bahwa penerapan teknologi berbasis Java dapat mendukung digitalisasi proses bisnis, khususnya di bidang pengelolaan laundry, untuk meningkatkan efisiensi dan akurasi operasional.